Dialog Boxes

Dialog Boxes

- Popup child windows created by Windows
- Used for special-purpose input & output
  - Principal I/O mechanism in Windows
- Contain several child window controls
- Layout & what it does is are predefined (template--a resource)
- How it does is determined by a "Dialog box procedure"
- Destroyed immediately after use

Types of Dialog Boxes

- Modal
- Modeless
- System Modal

WM_INITDIALOG Message

- Start Dialog box with call to DialogBox(…)
  - Causes WM_CREATE & WM_INITDIALOG msgs
  - WM_INITDIALOG is like an ordinary window's WM_CREATE message, but after controls have been ceated
- Processed before window (dialog box) is made visible
- Good place to put dialog box initialization code
- In an MFC CDialog-derived class, this message activates dialog box's *OnInitDialog()* handler

EndDialog(…)

- **Destroys dialog box**
- **Returns control to function (*WndProc()*) that started the *DialogBox()***

User Interaction with Dialog Box Controls

- **WM_COMMAND message**
  - **LOWORD(wParam) contains control ID**
  - **lParam, wParam contain message data**

Exchanging Data with a Dialog Box

- **Exchanging data between dialog box function and app's *WndProc()***
- ***SendMessage()* could be used to send message to control inside, BUT:**
  - **Need to know control's handle**
  - **Not known since Windows creates the controls**
  - **IDs are known--specified in resource template**
- **Use *GetDlgItem()* to get control's handle:**
  - ***hControl = GetDlgItem(hDlg, controlID);***
- **Then *SendMessage(hControl, Msg, wParam, lParam);***

Dialog Boxes in MFC

- MFC Dialog boxes are based on the CDialog class

Important MFC CDialog Functions

- *DoModal()* to start dialog box modally
- CDialog provides three over-rideable functions to initialize and respond to OK and Cancel button clicks
- *OnInitDialog()*
  - Handler for WM_INITDIALOG message
- *OnOK(), OnCancel()*
  - Handlers for WM_COMMAND messages from OK and Cancel buttons
  - Both call CDialog’s *EndDialog()* function to dismiss the dialog box and return control to *DoModal()*

Steps in Using a Modal Dialog Box (MFC):

- **1. Set up the dialog box template in the resources (.rc file)**
  - **Specifies controls used, their style/layout**
  - **Can be prepared "visually" with Visual Studio dialog box editor**
  - **Or "manually" with a text editor**
- **2. Create a CDialog-based class**
- **3. Instantiate a CDialog object**
- **4. Call its *DoModal( )* function**

Using Modal Dialog Boxes in MFC

- Dialog boxes are encapsulated by CDialog class (derived from CWnd)
- <u>2. App derives its own dialog box from CDialog</u>
  - e.g., *class CMyDlg : public CDialog*
    - Constructor should specify that parent constructor will be used
    - Also ID of DBox resource template to be used (IDD_XXX)
  - Dialog box msg handling done w/ message maps
  - Dialog box class declarations (.h file):
    - Message map and handling function declarations
  - Dialog box class implementation (.cpp file):
    - Message map and handler function definitions
  - Use Class Wizard to generate the CDialog-based class
    - Sets up msg mapping, constructor & correct Dbox resource ID

- <u>3. App instantiates the Dialog Box:</u>
  - Usually done in CView class in response to a main window menu item selection
  - CMyDlg dlg;
    - Creates the dialog box (not activated yet)
    - Initialization code, if any, should be put in CDialog's *OnInitDialog()* handler function
      - Invoked in response to WM_INITDIALOG message

✍ 4. Activating the Dialog Box

- Use CDialog's ***DoModal()*** member function
  - dlg.DoModal();
- Displays the dialog box
- Messages from dialog box controls go to dialog box handler functions
- Continues until dialog box has been closed by user clicking OK or Cancel buttons
  - CDialog's *EndDialog()* member function causes *DoModal()* to return
  - Can test return value
    - If(dlg.DoModal()==IDOK {//do something}
  - Message processing continues in parent window

Communicating with Dialog Box Controls (exchanging data)

✍ Method 1

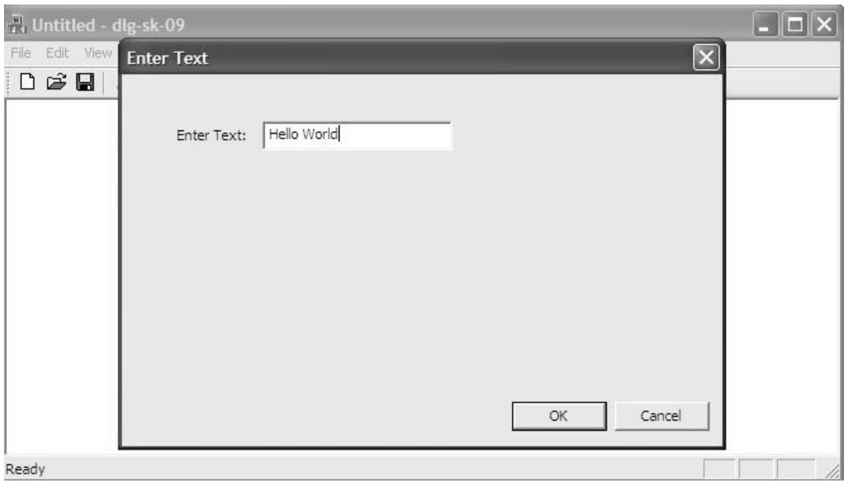
- Get a pointer to control's ID w/ CWnd::GetDlgItem()
- Use pointer to send appropriate messages to control, e.g. (for a list box in a dialog box):
  - *CListBox* pCtrl=(CListBox*)GetDlgItem(IDC_CTRL);*
  - *pCtrl->SendMessage(WM_GETTEXT,…);*
  - *GetDlgItemText(IDC_CTRL, m_string);* combines these two
    - m_string would be a pulbic variable to hold retrieved string
  - *SetDlgItemText(IDC_CTRL, m_string);*
    - Sends the string to the control
- OK for non-Wizard-generated apps
- There's a much easier way for Wizard-generated applications

- ✍ Method 2
  - – Use DDX (Dialog Data Exchange) mechanism
  - – Automatically built into Wizard-generated Apps
  - – DDX system moves data between dialog box controls and variables in Cdialog-derived class
  - – Occurs when a call is made to CWnd::UpdateData(direction);
  - – Boolean parameter sets direction of data movement
    - TRUE ✍ from controls to variables
    - FALSE ✍ from variables to controls

- ✍ MFC's *CDialog::OnInitDialog()* calls *UpdateData(FALSE)* automatically
  - – (Recall, this is called to start the dialog box)
    - So Data from program variables is transferred automatically to dialog box controls when the dialog box starts
- ✍ MFC's *CDialog::OnOK()* calls *UpdateData(TRUE)*
  - – (This is called when user clicks the "OK" button inside the dialog box)
    - So data from dialog box controls is transferred automatically to program variables when user clicks the dialog box's "OK" button)
    - *OnOK()* then calls *CDialog::EndDialog()*
      - – So dialog box disappears and *DoModal()* returns
        - – Returns IDOK or IDCANCEL depending on user action
      - – Destructor destroys the dialog box

Adding a Modal Dialog Box to the Sketching MFC Application

- Will allow the user to specify text to be displayed in parent window



Enter Text: Hello World

- Create a new Visual C++, MFC, SDI application (as usual)
- Add the sketching code (see earlier example)
- Add a new “Text” menu item (ID_TEXT)
- Add the new dialog box
  - Project/Add Resource/Dialog/New
  - Change ID to IDD_TEXT
  - Caption: “Enter Text”
- Use the dialog box editor to drag over a static and an edit control:
  - Static Control: “Text String”
  - Edit control: IDC_TEXTEDIT

- Create the new Dialog Class
  - Right click on an unoccupied area of the dialog box & choose “Add Class” to bring up the “MFC Class Wizard” Dialog Box
  - Class name: “CTextDlg”
  - Base class: “CDialog”

- ✍ **Add New Class Variables (and connect to controls):**
  - **In Class View, right click on CTextDlg & choose Add variable**
    - **In resulting "Add member variable Wizard"**
      - **Check "Control Variable" check box**
      - **Control ID: IDC_TEXTEDIT**
      - **Category: Value**
      - **Variable type: CString**
      - **Variable name: m_text**

- ✍ **Add handler code to new CView "Text" menu item**
  - **In Class View select CView-derived class**
  - **In Properties Wizard Box "Events" (lightning bolt icon):**
    - **Scroll down to ID_TEXT**
    - **Add Command handler *OnText()***
    - **Edit the resulting code by adding:**

      CTextDlg dlg;
      dlg.DoModal();
      pDC = GetDC(); // Assumes a CDC* pDC variable
      pDC -> TextOut(0, 0, dlg.m_text, lstrlen(dlg.m_text));
- ✍ **At top of Cview .cpp file underneath the other include statements, add:**
- ✍ **#include TextDlg.h**